Shake Them Up! A movement-based pairing protocol for CPU-constrained devices

Claude Castelluccia, Pars Mutaf

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Shake Them Up!

A movement-based pairing protocol for CPU-constrained devices

Claude Castelluccia — Pars Mutaf

N° 5457

January 2005

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--5457--FR+ENG

INRIA
RHÔNE-ALPES

Shake Them Up!
A movement-based pairing protocol for CPU-constrained devices

Claude Castelluccia*, Pars Mutaf †

Thème COM — Systèmes communicants
Projets Planete

Rapport de recherche n° 5457 — January 2005 — 28 pages

Abstract: This paper presents a new pairing protocol that allows two CPU-constrained wireless devices to establish a shared secret at a very low cost.

Our scheme requires that the devices being paired, A and B, are shaken during the key exchange protocol. This is to guarantee that an eavesdropper cannot identify the packets sent by A from those sent by B. A can then send the secret bit 1 to B by broadcasting an (empty) packet with the source field set to A. Similarly, A can send the secret bit 0 to B by broadcasting an (empty) packet with the source field set to B. Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on a n-bit secret key.

This paper presents the details of the protocol and the results of some experimentations. To our knowledge, this is the first practical pairing scheme that does not rely on expensive public-key cryptography, out-of band channels (such as a keyboard or a display) or specific hardware. The proposed protocol has very small computation and storage requirements. It is therefore well adapted to CPU-constrained devices (such as sensors) that have very limited capacities and are easy to shake.

Key-words: wireless devices, sensors, key exchange, pairing protocol

* INRIA Rhône-Alpes, PLANETE group
† INRIA Rhône-Alpes, PLANETE group

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Shake Them Up!
A movement-based pairing protocol for CPU-constrained devices

Résumé : Ce rapport présente un nouveau protocole qui permet à deux terminaux sans-fil A et B avec des capacités très limiées de d'échanger une clé secrète.

Note protocole nécessite de secouer les terminaux pendant la phase d'échange de clé afin de rendre l'identification de la source des paquets difficiles. A peut alors envoyer le bit 1 à B en diffusant un paquet dont le champs "adresse source " est positionnée à A. Il peut envoyer le bit 0 en diffusant un paquet dont champs "adresse source " est positionnée à B. Seul B peut identifier la source réel du paquet (étant donné qu'il n'a pas envoyé le paquet, la source est A), et recupérer le bit secret (1 si le champ source est positionné à A, 0 autrement). En répétant ce protocole n fois, une clé de longueur n bits peut être échangée entre A et B.

Ce rapport présente les détails du protocole et des résultats expérimentals. D'après nos connaissance, il s'agit du premier protocole d'échange de clé qui n'utilise pas la cryptographie ou des canaux sécurisés.

Mots-clés : capteurs, échange de clés, sécurité

1 Introduction

The current trend in consumer electronics is to embed a short-range wireless transmitter and a microprocessor in almost everything. The main motivation is to facilitate communication and cooperation amongst wireless devices in order to reduce their size/cost and increase their functionalities. In this context, each device can be seen as a peripheral of the others. For example, a user can use the display and the keyboard of a PDA to access his cellular phone or personal server [16]. Similarly, a user can use its cellular phone or PDA to retrieve temperature data sensed by a local sensor [15].

The main security challenge is to securely associate the devices together. For example, when a device receives data from a sensor, it needs to make sure that the data is received from the sensor it has selected and not from an impostor. Furthermore, integrity and privacy are often very important too.

The process of securely associating two wireless devices is often referred to as *pairing*. This process allows two devices, communicating over a short-range radio, to exchange a secret key. This key can then be used to authenticate or encrypt subsequent communication. It is important to notice that the key exchanged in a pairing protocol does not need to be authenticated since the identities (often provided by certificates) do not matter in this context. A user who is pairing two devices together only needs assurance that a key was exchanged between the devices he/she has selected (for example, the two devices he/she is holding in his/her hands).

In summary, a pairing protocol is composed of two separate sub-protocols:

1. *Key exchange* sub-protocol: this protocol is run between the two wireless devices and results in a secret key shared between the two devices.

2. *Pairing validation* sub-protocol: this protocol is executed between the two wireless devices and the user. Its goal is to guarantee (with some large enough probability) to the user that a key was exchanged between the two devices he/she actually wished to pair.

Motivations and design constraints: The motivation of this work is to design a pairing protocol for CPU-constrained devices, such as sensors. Designing pairing protocols for such environment is very challenging because sensors have limited CPU and memory. Also, because of their low costs, most of them cannot rely on tamper resistant components. The consequence of the limited computing and storage capabilities is that modular arithmetic is difficult and therefore, asymmetric cryptography cannot be used. In particular, standard Diffie-Hellman (DH)[4] key exchange protocols are excluded. Even low exponent RSA[14] techniques that allow to minimize encryption cost are prohibitive when sensors are involved. Our goal is to design a pairing protocol that meets these constraints.

More specifically, we aim at designing a protocol that does **not** use public key cryptography and does not rely on some preconfigured information. Furthermore, the designed protocol must not increase the complexity and the cost of the sensors by requiring additional hardware (such as a display, an I/O interface or an out-of band channel, such as an

infrared one). Finally, it should not require exotic wireless technologies, but instead work with current wireless networking standards such as 802.11 or 802.15.4 (an emerging Wireless PAN (WPAN) technology, designed for low power sensors). The proposed protocol must be secure against passive and active attacks. In other words, it must not allow active or passive attackers to learn the key exchanged between two paired devices. It must provide protection against Man-in-the-Middle (MitM) attacks that attempt to impersonate one or both of the devices during key agreement (i.e., how to make sure that the key is exchanged with the correct device?) It must also provide some protection against Denial-of-Service (DoS) attacks, i.e. prevent attackers from disrupting the pairing protocol and from exhausting the devices' resources, such as their battery.

Contributions: We present a novel secure pairing technique based on a key agreement protocol that does not depend on CPU-intensive operations. Two CPU-constrained wireless devices A and B can establish a shared secret over an anonymous broadcast channel. An anonymous channel is a channel on which an eavesdropper can read the packets that are exchanged but is unable to identify the source. Using such a channel, A can send the secret bit 1 (resp. 0) to B by broadcasting an (empty) packet with the source field set to A (resp. B). Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on a n-bit secret key.

The protocol is secure if and only if the packets of A and B cannot be distinguished by an eavesdropper. On a wireless channel, this property is difficult to achieve through protocol design since an eavesdropper can measure the signal strength of each packet and may be able to determine the real source of each packet. Therefore, we propose that during key agreement, the user(s) be very close to each other and shake their devices (i.e. randomly turn and move one around the other) in order to randomize the reception power of their packets by a potential eavesdropper and make power analysis very difficult.

Organization: The paper is structured as follows: Section 2 presents the related work. Section 3 presents the basic ideas of our scheme. Section 4 describes our proposal in detail. Section 5 presents experimental results and analysis. Finally, Section 6 concludes the paper.

2 Related work

The problem of secure pairing of wireless devices has been tackled by several researchers and the proposed solutions can be classified into three main categories:

1. *Public-key cryptography based solutions*:

   These solutions rely on public-key based key exchange protocols such as Diffie-Hellman or RSA [5, 9, 10]. In Diffie-Hellman based schemes, devices exchange their Diffie-Hellman components and derive a key from them. In RSA-based schemes, one of the devices selects a secret key and encrypts it under the other device's public key.

The main problem of these solutions is performance. They require that devices perform very CPU-intensive operations such as exponentiation, which are prohibitive for CPU-constrained devices.

2. *PIN-based schemes*:

   In Bluetooth, two wireless devices derive a shared key from a public random value, the addresses of each device and a secret PIN number. The PIN number is provided to each device by the user via an out-of-band channel, such as a keyboard. While this solution is computationally efficient, it requires that *both* devices be equipped with some kind of keyboard or input interface. As a result, this solution cannot be used to pair devices lacking input interfaces, such as sensors.

3. *Physical contact or imprinting*:

   In [15], Stajano and Anderson propose a solution based on physical contact. Two devices get paired by linking them together with an electrical contact that transfers the bits of a shared secret. No cryptography is involved, since the secret is transmitted in plaintext. Furthermore, the key validation phase is not necessary since there is no ambiguity about the devices that are involved in the binding (i.e. MitM attacks are impossible).

   While this solution is interesting, it requires to equip each device with some additional hardware to perform the electrical contact. Similarly, it might be possible to transmit a secret key through an infrared channel to a nearby node. Infrared transmissions require absolute line-of-sight links, making it more difficult for third-party interception. Nevertheless, in both cases, i.e. physical contact or infrared transmission, the complexity and the cost of the devices would increase [1]. This violates one of our design requirements, that the pairing protocol should not require extra equipment. We wish to achieve key agreement through an existing channel which may also be used for communication.

Note that all the previous schemes (except the last one) require some additional mechanism to validate that the pairing was performed between the two intended devices (pairing validation). The only solution proposed in the literature so far is to provide to the user some evidence that both devices computed the same secret key. For example, the devices can both display a hash of the secret key [5]. These solutions are not always practical since they require devices with a display and/or a keyboard.

3 Basic ideas

This section describes the main ideas of our scheme. We first describe how two devices, communicating over an anonymous channel, can exchange a secret key without expensive

[1] Since infrared channels require line-of-sight links, they cannot be efficiently used for the actual communication between the sensors.

computation. We then define formally what we mean by anonymous channels and describe how they can be implemented in practice.

3.1 Pairing over anonymous channels

This section describes a new technique that allows two parties to securely exchange a secret over an anonymous channel while preventing eavesdroppers from determining its value (or actually any of its bits). By anonymous channel, we mean a broadcast channel that hides the origin of the messages. On an anonymous channel, a passive wiretapper can read the message that are broadcast but is unable to identify the source. Anonymous channels require a property that we call *Source Indistinguishabilily*. This property is defined and discussed in Section 3.2.

Our key exchange protocol was inspired from the protocol proposed by Alpern and Schneider in [1]. In this paper, the author presents a protocol that allows two parties to agree on a secret key on channels for which an eavesdropper cannot tell "who" broadcasts each message. The technique is called "Key exchange using keyless cryptography", or "Keyless key agreement".

For users Alice (A) and Bob (B) to agree on a n-bit key $K_{AB}[n]$, each first chooses its own random 2n-bit string:

$$R_A[1], R_A[2], ..., R_A[2n]$$

$$R_B[1], R_B[2], ..., R_B[2n]$$

User A then broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_A. Similarly, user B broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_B. The secret key is then defined by the bits $R_A[j]$ sent by A such that $R_A[j] \neq R_B[j]$. Note that there are on the average n such bits. The salient property of the protocol is that the message content is not hidden. All messages are accessible to potential eavesdroppers, which however cannot determine the origin of each message. As a result, they are unable to identify the packets sent by A and therefore identify the correct bits. Since A knows her bits, she can easily identify the bits sent by B. Similarly since B knows his bits, he can easily identify the bits sent by A. Note that the packets transmitted by A and by B must be interleaved. Otherwise it might be easy for an eavesdropper to identify the bits sent by a same source from a timing analysis. $R_A[1]$ and $R_B[1]$ should be sent first, followed by $R_A[2]$ and $R_B[2]$, and so on. Of course the transmission order must be random. In fact if A always sent first, an eavesdropper can easily recover the key.

The protocol presented by Alpern and Schneider requires the broadcast of $4n$ messages for A and B to agree on a n-bit secret key. We propose an optimization that reduces the number of broadcast messages to n. The overview of our protocol is the following (a more detailed description is presented in Section 4):

1. A selects n/2 random bits $R_A[1], R_A[2], ..., R_A[n/2]$
2. B selects n/2 random bits $R_B[1], R_B[2], ..., R_B[n/2]$

3. A builds n/2 messages $m_A[1], m_A[2], ..., m_A[n/2]$, where the source identifier of $m_A[j]$ is either set to A if $R_A[j] = 1$ or set to B if $R_A[j] = 0$.

4. B builds n/2 messages $m_B[1], m_B[2], ..., m_B[n/2]$, where the source identifier of $m_B[j]$ is either set to B if $R_B[j] = 1$ or set to A if $R_B[j] = 0$.

- A and B send their messages synchronously but in a random order. In other words, the first messages of A and B are sent (in a random order), followed by the second messages (in a random order), and so on. In total, n messages, m_x, are sent.

- For each message m_k that A (resp. B) receives, it checks whether the source identifier is set correctly (note that only A and B can perform this verification) and sets $K_{AB}[k]$ to 1 if the source is correct or to 0 otherwise.

3.2 Source indistinguishability: definition and requirements

The described key exchange protocol requires the *source indistinguishability* property. In other words, if two parties, A and B, run the previously described key exchange protocol, an eavesdropper should not be able to distinguish the packets sent by A from the packets sent by B. Failing to achieve this property actually leads to an insecure protocol, since the eavesdropper could then recover some (if not all) bits of the exchanged key.

This notion of source indistinguishability is very similar to the notion of ciphertext indistinguishability in encryption schemes [7]. The basic idea behind indistinguishability of an encryption scheme is to consider an adversary (not in possession of the secret key) who chooses two messages, m_1 and m_2, of the same length. Then one of the messages is encrypted and the ciphertext is given to the adversary. The encrypted scheme is considered secure if the adversary cannot tell which of the two messages was encrypted.

We define *source indistinguishability* in a similar way as follows: a communication scheme between two parties A and B is *source indistinguishable* if for a given packet P, emitted by A or B, an eavesdropper cannot tell whether the packet was sent by A or B. More formally, the difference between the probability that the packet was sent by A and the probability that the packet was sent by B should be very small:

$$Pr[source(P) = A] - Pr[source(P) = B] < \epsilon$$

In practice, source indistinguishability requires the communication to be *temporally* and *spatially* indistinguishable. In the following sections, we discuss these requirements in detail [2].

[2] We assume that packets do not carry information that can help identify the source address. Thus we concentrate our efforts on temporal and spatial indistinguishability problems.

3.2.1 Temporal indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *timing analysis*, to identify the packets emitted from A from those emitted from B with a probability larger than $1/2$. Furthermore, the eavesdropper should not be able to group packets emitted by the same source.

It is clear from the previous definition that a communication system that uses a TDMA (Time Division Multiplexing Access) based MAC (Medium Access Control) protocol cannot provide temporal indistinguishability. In a TDMA-based system, each terminal is given one or several time slots and can transmit only during one of its slots. As a result, it is very easy for any eavesdropper to identify the packets transmitted by a source or at least identify the packets sent by a same source.

Random access MAC protocols, such as CSMA (Carried Sense Multiple Access) are more appropriate. CSMA protocols such as Ethernet or wireless Ethernet, multiple nodes are allowed to use the same channel in a random fashion. Before transmitting data a node listens to the channel. If the channel is busy then it waits for a random time and then listens again. If the channel is not busy, then it transmits its packet. In CSMA, collisions may happen when two terminals transmit simultaneously. CSMA/CD (Collision Detection) enables devices to detect a collision. After detecting a collision, a device waits a random time period and then attempts to re-transmit its message. As we will show in Section 5, with a CSMA-based system, the order of the packets sent by the different users can easily be randomized. This feature is crucial for the security of our approach. In this case, it will be very difficult for an eavesdropper to use timing information to identify the source.

3.2.2 Spatial indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *spatial analysis* (or signal strength analysis), to distinguish the packets emitted by A from those emitted by B with a probability larger than $1/2$. In other words, the eavesdropper should not be able to detect the packets' source from their reception power.

This property is very difficult to achieve in practice since wave attenuates according to the *free space propagation* law and the eavesdropper can easily identify the location of the transmitter from the reception power of a received packet, i.e. from *power analysis*. More specifically, according to free space propagation law, the reception power (in Watts) Sp of a packet transmitted with power St by a transmitter that is located at a distance d is defined as:
$Sp = St * Gt * Gr * K * 1/d^2$, where Gt is the antenna gain of the transmitter, Gr is the antenna gain of the receiver and K is a constant that depends on the signal frequency (or wavelength). If the receiver knows St and the gain, it can easily estimate d. An eavesdropper listening to a communication between two parties A and B can also use the reception power to identify the source of the packets with a probability larger than $1/2$.

Let's assume that A and B use the same type of antenna (i.e. they have the same gain) and let's define $k = Gt * Gr * K$. If A and B transmit their packets with a power

uniformly distributed between $[St; St + delta]$ then A receives the packets from B with a power uniformly distributed in $[k.St/d^2; k(St + delta)/d^2]$. Similarly, B receives the packets from A with a power uniformly distributed in $[k.St/d^2; k(St+delta)/d^2]$. If the eavesdropper is listening with a large antenna (i.e. Gr' is large s.t. $k' = Gt * Gr' * K > k$) and it is closer to A than to B (i.e. $d_A \leq d_B$) then:

1. the power of A's packets received by the eavesdropper is uniformly distributed in $[k' * St/d_A^2; k' * (St + delta)/d_A^2]$ and,
2. the power of B's packets received by the eavesdropper is uniformly distributed in $[k' * St/d_B^2; k' * (St + delta)/d_B^2]$.

Therefore the eavesdropper can identify all the packets received with a power in $[k' * St/d_B^2; k' * St/d_A^2]$ as belonging to B and all the packets received with power in $[k' * (St + delta)/d_B^2; k' * (St + delta)/d_A^2]$ as belonging to A. The scheme can only be secure if of one the two following conditions is met:

1. *Condition 1*: $d_A = d_B$. The scheme is secure because the power of the packets sent by A is statistically indistinguishable from the power of the packets sent by B. If $d_A \neq d_B$, the eavesdropper can identify some of the bits of the secret. The number of bits that can be identify depends of the values of d_A and d_B. If $d_B > (St+delta/St)^{1/2} * d_A$, the eavesdropper can guess the source of all the packets exchanged between A and B and therefore all the secret. If $d_A < d_B < (St + delta/St)^{1/2} * d_A$, the eavesdropper can guess the source of some percentage of the packets. This percentage depends on the difference between d_A and d_B. Note that if the eavesdropper can monitor at several locations, and receive the same packets with different reception powers, it will be even easier for her to identify the source of the packets.
2. *Condition 2*: A and B should move during the pairing phase such that d_A and d_B (and therefore their respective powers) are statistically indistinguishable. This is the approach that we are alluding to in our scheme.

4 Movement-based pairing

This section describes a new protocol that can be used to pair two devices A and B securely and without using expensive public-key cryptographic protocols. Our key agreement protocol is based on the combination of the two following novel and original ideas: we first optimize Alpern and Schneider's keyless key agreement protocol and adapt it to an ad hoc configuration. Secondly, we show that the protocol can be secured against power analysis by shaking the two devices around each other. We show that it is secure against DoS (Denial-of-Service) and MitM (Man-in-the-Middle) attacks.

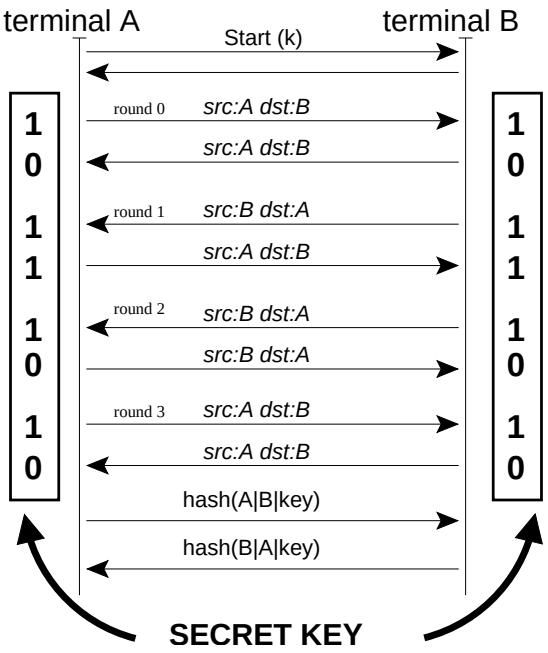


Figure 1: Key agreement protocol for movement-based pairing.

4.1 Key agreement

In our pairing protocol, key agreement is performed using the protocol described in Figure 1. This protocol is an optimization of the approach proposed by Alpern and Schneider [1]. The number of messages per secret bit is reduced (1 instead of 4), making the protocol more energy efficient.

The protocol starts by a START message transmitted by one of the two parties, either A or B. This message contains the value k, which is the number of rounds the protocol should execute, and the address of the packet's source. Upon reception of this message, the other party replies with another START message that contains its address. This exchange allows each device to learn the other party's address and the size of the desired shared key. It should be trigger by the user, by, for example, pushing a button of the devices.

At each round j, A broadcasts its (empty) packet at time $t_{A,j}$, where $t_{A,j}$ is randomly selected in the interval $[(j-1)*T, j*T]$, where T is a constant. Similarly, B broadcasts its packet at time $t_{B,j}$, where $t_{B,j}$ is randomly selected in the interval $[(j-1)*T, j*T]$. As a result, the order of transmission of these two packets is random in the interval $[(j-1)*T, j*T]$. An eavesdropper is then unable to detect the source of the packets using timing information. Secret bits are represented by the correct or inversed placement of source and destination addresses. If the sender and recipient addresses are correct the terminals A and B presume a secret bit TRUE (1), otherwise they presume FALSE (0). For example, in Figure 1, the first message is sent by A. Furthermore, the source address is set to A and the destination address is set to B. Since the packet was actually sent by A, the resulting bit (the first bit) of the secret key is then set to 1 by A and B. Note that an eavesdropper cannot identify the real source of the packet and therefore cannot identify the value of the exchanged secret bit. Each message identifies one bit of the secret key. At the end of the k rounds, A and B share a $2*k$-bit long secret key. Therefore, if a 60-bit long key is required, the protocol should contain 30 rounds, i.e. 60 messages.

The protocol is terminated by two messages, that are used to validate the exchanged key. The message sent by A contains the value $a = hash(A|B|key)$, where key is the exchanged key. The message sent by B contains the value $b = hash(B|A|key)$. The order of these two messages is also random. When B receives the value a, it can verify that A has the same key. Similarly, A can verify, upon reception of b, that B computed the correct key.

4.2 Achieving spatial indistinguishability: Shake them up!

As described in Section 4, the previous scheme is secure only if the source of the packets are indistinguishable.

Time indistinguishability is provided by randomizing the order of transmission of packets sent by A and B, as described in the previous section. An eavesdropper can therefore not guess who is going to transmit next. Also, we are using CSMA-based wireless systems, such as 802.11, to guarantee that the access to the channel is also random and does not reveal any information about the source.

As explained previously, *spatial indistinguishability* is more difficult to achieve. We propose to achieve this property with user assistance. The user(s) should shake (i.e. move and turn) the devices during key agreement in order to equalize the average signal strength of the two devices measured by a potential eavesdropper.

The required movements depend on the type of antennas used. For truly omni-directional antennas, antenna orientation will not reveal any signal strength difference between two devices. In these cases, it will be sufficient to take both devices and turn them one around the other, in order to equalize the effect of distance (between each terminal and the eavesdropper) on the signal strength measurements performed by an eavesdropper. If the antennas are not truly omni-directional, randomizing the distance will not be enough to achieve spatial indistinguishability. Different orientation of the devices may reveal a serious signal level difference. In order to avoid this problem, during key agreement the two devices must be randomly turned to different directions (in our experimentations we used commodity 802.11 cards which are not omni-directional, therefore Section 5 contains much more detail on this issue).

Clearly, in both cases, having small and lightweigth devices (e.g. sensors or small PDAs) will reduce the user burden for key agreement. The user can take one device in each of his hands and randomly move them one around the other according to the horizontal and vertical axes. If the devices are very small, he can take both of them in one hand and shake his arm, like he would do with an orange juice bottle.

The security of the proposed scheme depends on the quality of the movement. Users of our scheme should be aware that it is their responsibility and in their best interest to move the devices properly during the pairing phase. Note that movement-based operations or "protocols" are quite frequent and accepted in our everyday lives. For example, orange juice or shaving cream bottles are universally shaken prior to usage. This is now a well-known and quite a natural protocol. Furthermore it is commonly accepted that it is the responsibility and interest of the consumers to perform this shaking operation properly.

4.3 Protection against MitM and DoS attacks

4.3.1 Protection against MitM attacks

Defeating a MitM (Man-in-the-Middle) attack requires assurance for a terminal A that a secret key is really being exchanged with the intended terminal B and not an impostor's device. This is the goal of the *Pairing Validation* protocol, as described in Section 1.

In our case, this problem is reduced to the following issue: "how the two devices reliably determine each other's address in a communication channel where many devices are present?" This is a more general question which may have different answers in different systems.

Once the two devices obtain each other's address correctly, MitM attacks will be impossible in our scheme. This is mainly because the devices are very close to each other (which is required for spatial indistinguishability) and hence they can easily detect maliciously inserted messages that impersonate them. Also, the short distance between the devices will help the devices to reliably determine each other's address using signal level measurement.

I.e. it can be required that "START" messages be received with a high signal level. A distant impostor may attempt to foil this technique by increasing its transmission range. In this case, however, the "START" messages from the same device will be received at a higher rate (i.e. bogus ones mixed with the real ones). Such anomalies can be easily detected in our case.

One of the devices, say device B, may be down and an impostor may profit from the situation by imitating B to A. To our knowledge such an attack has no solution if we do not use CPU-intensive cryptographic techniques (that we wish to avoid in this paper). The devices must probably inform the user about their status, i.e. up or down, using for example a periodic led signal.

4.3.2 Protection against DoS attacks

We can differentiate between two kinds of DoS (Denial-of-Service) attacks on a key agreement protocol. In the first one an attacker may exploit the key agreement protocol to force a victim to perform computationally expensive operations, with the goal of draining its battery or preventing it from performing useful work. Unlike public-key cryptography-based schemes, our protocol is not based on CPU-intensive operations and therefore immune against such DoS attacks. Another DoS attack may consist of sabotaging the key agreement, i.e. making it impossible for the two parties to agree on a same secret key. Our basic protocol illustrated is Figure 1 is vulnerable against such attacks and in this section we describe a solution.

In the protocol illustrated in Figure 1, it is very easy for an attacker to insert a bogus packet with source address A and destination address B (or vice versa) and perform what we call a *key poisoning* attack. Such a packet inserted by a third party would generate different secret bits at the terminals A and B. The attacker can insert an arbitrary number of bogus packets, and make it impossible for A and B to agree on a secret key.

The protocol depicted in Figure 4.3.2 defeats the key poisoning attack. In this protocol, each secret bit is constructed using one packet from A and another from B, i.e. both terminals contribute to the construction of each secret bit [3]. Each secret bit is given a sequence number (which also corresponds to the round number). In order to generate the secret bit $\#i$ the two terminals generate a packet with correct or flipped address positions, in random order (that is the probability that the first packet $\#i$ will be transmitted by A is 0.5). The result of the two packets $\#i$ are combined by taking their sum (mod 2), or exclusive OR. The result is the secret bit $\#i$.

To corrupt the bit $\#i$, an attacking node can insert x packets with the same sequence number, where $x = 1, 2, 3, ...$ In this case both sides will note $x + 2$ bits with the same sequence number, but only two of them will be the same at both sides. Let $\{a_1, ..., a_{x+2}\}$ and $\{b_1, ..., b_{x+2}\}$ the set of bits (with the same sequence number) that the terminals A and B have agreed upon. Then we have

[3] This protocol requires twice as many messages as the basic protocol. However, this protocol is only necessary when the user believes that his devices are under DoS attacks. Otherwise, the basic scheme should be used.

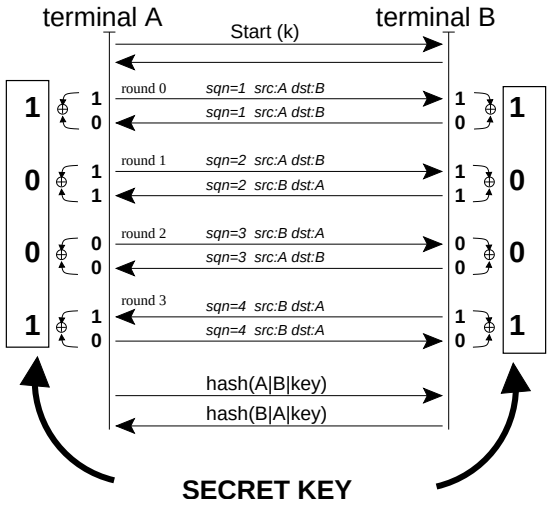


Figure 2: A protocol that resists what we call a *key poisoning DoS attack*. When this protocol is used for key agreement, an active attacker cannot poison (i.e. corrupt) the secret key by inserting bogus messages with the addresses of A and B (see text for details). This protection is obtained at the cost of two messages per secret bit.

$$a_1 \oplus \cdots \oplus a_{x+2} = b_1 \oplus \cdots \oplus b_{x+2}$$

if x is pair, and

$$b_1 \oplus \cdots \oplus a_{x+2} \neq b_1 \oplus \cdots \oplus b_{x+2}$$

if x is odd.

Key poisoning can be defeated by taking the sum (mod 2) of every bit with the sequence number i, as usual (except that in normal operation $x = 0$). If the attacker inserted an odd number x of packets, the terminal A must invert the resulting secret bit $\#i$. In these conditions, an attacking node cannot poison the key shared by A and B by inserting bogus secret bits.

Note that this protocol fails when A and B do not receive the same messages. This might happen when some of the messages are lost. We therefore suggest that A and B append to each of their messages a hash of all the previous messages they have seen since the beginning of the protocol. As a result, if one or several messages are lost, A and B can detect it immediately (instead of waiting until the end of the protocol and comparing a hash of the derived keys).

5 Experimentations and analysis

In this section, we analyse, by experimentations, the security of the proposed pairing scheme. More specifically, we show that signal power and timing analysis can not be used by an attacker to retrieve the key exchanged between two devices that use our pairing protocol. We also evaluate the energy cost of our protocol and show that eventhough it requires several messages, it is much more energy-efficient than a Diffie-Hellman based pairing protocol.

5.1 Setup and methodology

We built a testbed with lightweight laptops equipped with a PCMCIA Lucent IEEE 802.11 Wavelan card operating at 2.457Ghz (802.11 channel 10) and 11Mbps bit rate and in ad hoc mode. Our cards support RSSI (Received Signal Strength Indicator) and allow us to visualize and evaluate the power of received packets. Our signal level cryptanalyzer is built upon Linux wireless tools[4], and in particular *iwspy* that allows to get per node link quality. The *iwspy* command takes as argument a MAC address M, and outputs the received signal and noise levels of packets whose source address is M. Figure 3 depicts a typical measurement that can be carried out by any user using the *iwpsy* tool and a simple sampling script. Note that *iwspy* also outputs the noise level which is about -96dBm in our environment.

The received signal strength depends on at least 3 distinct factors:

1. Transmission power: all packets in our experiments are transmitted at our cards' default value which is 15 dBm[5].
2. Distance between the source and the signal level analyzer.
3. The relative angle between the source and the signal level analyzer: the cards that we use are not omni-directional and the received signal level heavily depends on the relative angle of the two cards.

During each experiment, refered as 'scenario', Eve (eavesdropper, or cryptanalyzer) measures the signal strength of the packets sent by Alice (terminal A) and Bob (terminal B) during key agreement. Many different experiments were carried out and in this paper we provide the most representative ones that we consider generic and applicable to almost all situations because they perfectly reflect the points (2) and (3) listed above. Our first scenario, denoted *scenario1*, is illustrated in Figure 4. In this scenario Alice and Bob are close to each other (within 0.5 meter) and make two kinds of movements in order to equalize their average signal strength captured by Eve:

[4] Available at: http://www.hpl.hp.com/personal/Jean_Tourrilhes

[5] Note that the spatial indistinguishability property requires that the two devices set the same transmission power. We observed that almost all vendors set it to 15 dBm. In case of exceptions, the devices should modify their transmission power to a well-known value (that we do not precise in this paper) and keep it constant during key agreement.

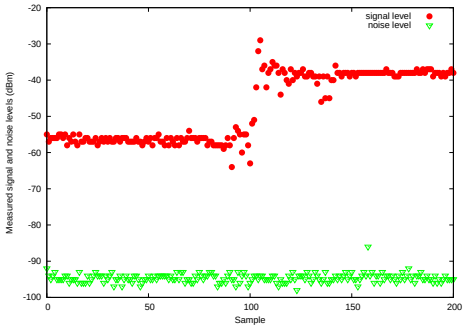


Figure 3: A typical *iwspy* output. In this example, the "iwspied" terminal is stationary while the first ~100 samples are taken and then it moves to a location that is closer to the signal level analyzer machine. The distance between the two cards is very well captured using signal strength analysis. Almost all modern wireless Ethernet cards support RSSI and Linux wireless tools are freely downloadable by any Internet user. Consequently, we note that the attack that we call *power cryptanalysis* is not only a theoretical threat; it can be easily mounted by any user with average system and programming skills.

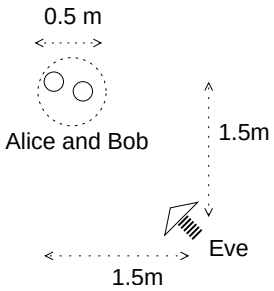


Figure 4: Scenario1

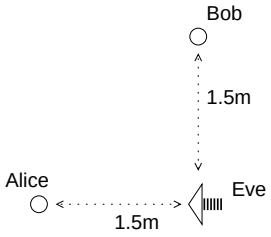


Figure 5: Scenario2

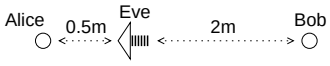


Figure 6: Scenario3

- We use commodity wireless Ethernet cards and they are not omnidirectional. Thus, in order to confuse Eve, Alice and Bob must turn their laptop in randomly changing directions (at a reasonable speed). The process requires reasonable effort from the user and takes around 16 seconds for agreeing upon a 80-bit secret key. The details of movement speed and its effect on security will be discussed later.
- Alice and Bob move their devices one around another with a reasonable effort, i.e randomly and at a moderate speed. This helps Alice and Bob to hide their relative distance between their cards and a potential eavesdropper that may be located anywhere.

In scenario1, we consider a pessimistic passive attack where Eve's wireless Ethernet card (the white arrow) is directly oriented to Alice and Bob and situated only 2.2 meters away. This allows Eve to make relatively accurate signal level measurements. In practice, Alice and Bob would probably notice the presence of a third person during key agreement, and look for another place where eavesdroppers cannot approach them. However, in some situations such countermeasures may not be practical. This scenario attempts to capture the cases where the presence of a third person cannot be avoided. Note also that an eavesdropper may have installed hidden signal level cryptanalyzers at strategical points. Thus, the absence of a third person, does not necessarily imply a secure environment. In this scenario Alice and BoB respect the key agreement requirements, hence the key will be secure as we will show below.

In scenario2 (Figure 5), we demonstrate an inappropriate usage of our protocol that we would like to disadvise. In this scenario Alice and Bob are not close to each other. They both move their laptop randomly in every possible directions, but they are always far from each other and their location does not change during key agreement. Eve profits from the distance between Alice and Bob, and directs her card to Alice. Consequently, Alice's packets are received at a higher signal level than that of Bob, rendering the secret key weak.

The scenario3 (Figure 6) is even less secure and firmly disadvised. Alice is 4 times closer to Eve than Bob. Eve is located between the two terminals and profits from the situation by directing her wireless Ethernet card to Alice. Altough Alice and Bob's cards are perfectly turned in random directions, Eve can easily diffentiate between their packets. As a consequence, the key will be extremely weak.

5.2 Security Analysis

This section provides a security analysis of our key-exchange protocol in the settings of the three scenarii presented previously. We first present the tools that we use for our analysis. We then apply these tools to demonstrate the secrurity or insecurity of our proposal against attackers that use power or timing analysis.

5.2.1 Method of Analysis

The security of the proposed scheme appears to be very similar to the security of systems, such as smart cards, against power and timing analysis [2]. We therefore propose to use similar security arguments.

If given two parties A and B and two collection of samples X_A and X_B that they emanate during the key exchange (X can power or timing information), the proposed scheme is secure (i.e. leakage-immune) if the variables X_A and X_B are statistically indistinguishable.

The intuition behind this definition is that if the two variables are indistinguishable, there isn't any algorithm that an attacker could potentially use to differentiate the packets of A from those of B. This definition is probably overly cautious, because in practice an attacker do not have access to X_A and X_B separately but rather gets an aggregated view of both of them. However, it seems difficult to come up with a less stringent definition that is still practical.

Evaluating the indistinguishability properties of two sets of samples X and Y can be performed using significant tests in statistics. Given two sequences of samples, a significance test returns the probability α that an observed difference in some featuresof X and Y could rise by chance assuming that that X and Y were generated by the same source. If the resulting probability is small (typically less than 1% [2]), then the difference between the two sequences is not due to chance but rather because the two sequences were generated by two different sources. In this case, the sequences are not indistinguishable. If the resulting α is larger than 1% then the two sequences are *possibly* indistinguishable. In fact, it might be misleading at this point to conclude that the two sequences are statistically indistinguishable, because further testing might reveal a genuine difference. However, by executing several different significant tests on the sequences, one might get reasonable evidence that the sequences are indistinguishable. In practice, 20 different tests should be applied before consider the sequences possibly indistinguishable.

In this paper, we access the security of our scheme using two of the most popular tests, namely the *distance of mean (DoM)* and the *sum of rank (SoM)* [6]. We use these tools to analyse the difference of power signals (power analysis) and of timing information (timing analysis) monitored by the attacker in the three scenarios described previously.

5.2.2 Signal power analysis

As described previously, an attacker could use the reception power of the packets it receives to differentiate the packets sent by A from those sent by B. Obviously, if the attacker

receives A's packets with a much larger (or lower) power, it can easily identify the source of each packet and our key exchange protocol becomes insecure.

In this section we analyse the signal power of packets sent by A and B in the three scenarii presented previously. During an pairing, each terminal sent 80 packets. Each packet is received by the attacker with a certain power. We denote P_A the sequence of the 80 values that represent the reception power of the 80 packets sent by A. Similarly, P_B defines the sequence of the 80 values that represent the reception power of the 80 packets sent by B. The sequence P_A and P_B for the above three scenarii are displayed in Figure 7.

- **Scenario1:** we observe that the powers Alice and Bob's packets are mixed and not easily distinguishable by Eve (the reader may imagine that Eve's vision has only one color regardless of the sender's ID). The only information available to Eve will be the absolute value of signal level difference between two packets captured during each round (1 packet from Alice, 1 packet from Bob). In Figure 8 we provide a frequency diagram of these signal level differences. It should be noted that, when plotting these histograms we have profited from additional information that is not available to Eve: the sign of the observed differences (i.e. (+) when Alice's packet is received with greater signal level than that of Bob, and (-) otherwise). These histograms were plotted using 1000 round experiments in order to provide accurate results that correspond to the average case (for a given scenario). Note also that, for our data collection purposes, Alice and Bob performed the required laptop movements for a much longer time than needed in practice: ~ 3.5 minutes for each experiment (in our experiments Alice and Bob generated 0.2 packets per second, as we will explain later). The histogram that corresponds to scenario1 is centered on $\sim$0 and roughly symmetric.

  We furthermore executed the DoM and SoR tests on sequences P_A and P_B. The results are summarized in Table 1. The values in this table indicate the probability α. The values in brackets indicate the value ϵ. Appendix A details how these values are computed with the DoM significance test. Details on how to compute these parameters for the SoR test can be found in **??**.

  The probabilities α computed with both tests in $Sce1$ are much larger than 1%. Therefore, the observed difference between the two sequences is *not significant* and can be attributed to chance. More tests are actually required to demonstrate that these two sequences are statistically indistinguishable. However, the results of these two tests are quite promising and we can probably conjecture that Alice and Bob's packets cannot be distinguished using signal strength analysis.

Table 1: Significance tests for Power Analysis

	Sce1	Sce2	Sce3
Distance of means	0.36(0.91)	0.0(9.745)	0.0(23.49)
Sum of ranks	0.87(0.157)	0.0(6.92)	0.0(11)

- **Scenario2:** In practice, the results look satisfactory in scenario2. There is no well defined technique (at least to our knowledge at time of writing) that will allow to clearly distinguish Alice's and Bob's packets. Note for example that, above the line -50dBm all packets are Alice's packets. Similarly, below the line -35dBm, we have only Bob's packets. However, unlike the reader, this information is not provided to Eve.

  On the other hand, the resulting key is clearly insecure in theory. As shown in Figure 7-b the signal level differences are important: the histogram is centered at 7.34 dBm. Although it is unknown to the attacker, this difference is considerable (making us uncomfortable) and reflects very well the fact that Eve's wireless Ethernet card is directed to Alice. Futhermore as shown in table 1, the probablity α is smaller than 1% in $sce2$, and therefore the sequences P_A and P_B are *significantly* different.

  In conclusion, eventhough we could not propose any practical attacks, we showed some evidence that both sequences are difference. We therefore disadvise the type of scenario where Alice and Bob are 'not' close to each other.

- **Scenario3:** In the final scenario, the situation is clearly worse. The resulting key is not only 'theoretically' breakable as shown by the frequency diagram (centered at 14.92 dBm), but also breakable in practice. Figure 7-c reveals what we call a "break point" which is situated around -41dBm. There is a visible gap at that point where Alice and Bob's packets are clearly separated. The results of the significance test show clearly that both sequences P_A and P_B are significantly difference. The protocol is obviously insecure in the settings of this scenario.

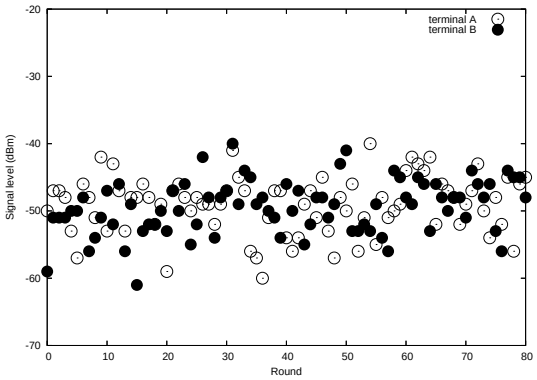
5.2.3 Timing analysis

Another attack that Eve can mount is what we call a *timing analysis* attack. The attack consists of measuring the time interval between each consecutive packet. Thus, care should be taken during the protocol design phase in order to avoid timing flaws and satisfy the temporal indistinguishability requirement described in Section 3.2. While this is essentially an implementation issue, it has bearings on the theoretic security of the protocol and therefore in this section we discuss this problem and describe solutions.

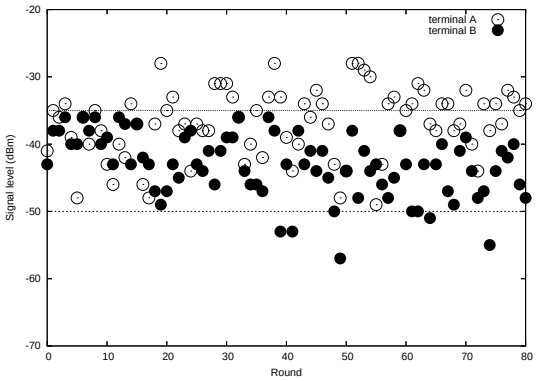
In order to defeat timing cryptanalysis attacks, Alice (and Bob) can insert a random delay between each of one her consecutive packets. In this case, assuming a good random number generator [6], Eve could not distinguish the source of each packet using timing analysis.

Another alternative that satisfies the temporal indistinguishability property is illustrated in Figure 9. This algorithm allows the packets to be transmitted at a constant rate, facilitating the user movements (to satisfy the spatial indistinguishability property against signal level cryptanalysis). In this algorithm two constants T and δ are defined, where T is a constant time interval before starting a round, and δ is the round duration. At the beginning

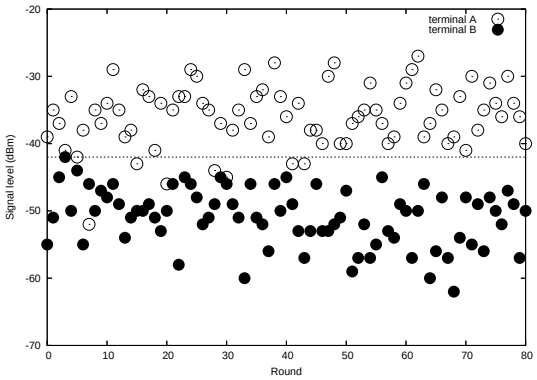
[6] A random number generator can be build from a hash function such as SHA1 at a very low cost. For example, a random number r_i can generated as $r_1 = SHA1(seed|i)$, where *seed* is a secret only known to the device and i is an counter that is increased by 1 each time a new random value is needed.



(a) scenario1



(b) scenario2



(c) scenario3

Figure 7: Received signal level of terminal A (Alice) and terminal B (Bob) during key agreement. The reader may imagine that Eve's vision has only one color (i.e. all packets are black).

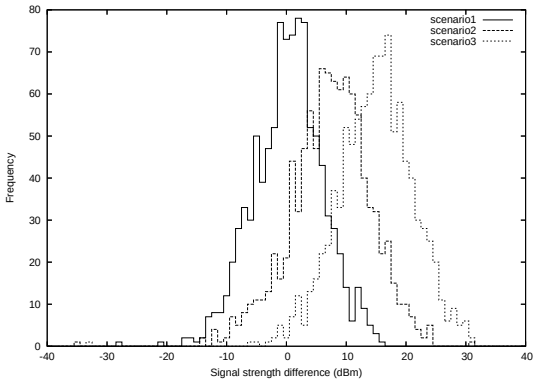


Figure 8: Frequency diagrams for signal level difference. In scenario1, the spatial indistinguishability requirement is satisfied. The histogram is centered on zero and symmetric. Thus, in this paper it is conjectured that an eavesdropper cannot distinguish the source of the packets regarding signal level difference (in scenario1).

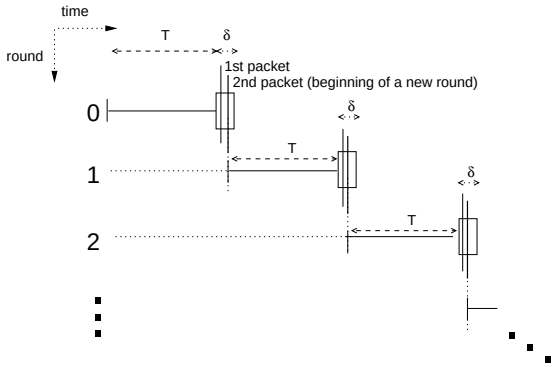


Figure 9: A timing algorithm that satisfies the temporal indistinguishability property, while allowing the packets to be broadcast at constant rate T. Each round takes δ time units or less. The first packet of each round is broadcast either by Alice or Bob (with 0.5 probability) which chose the smaller 'planned delay' (see text).

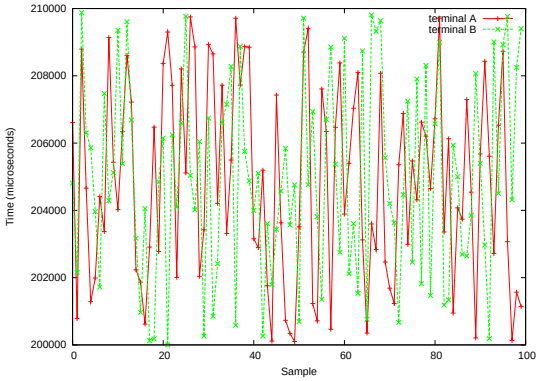


Figure 10: Time interval between messages sent by terminal A (Alice) and terminal B (Bob) during key agreement. The reader may imagine that Eve's vision has only one color (i.e. all packets are black).

of each round, the two terminals choose a different random value α that is uniformly distributed over [0-δ]. This random value can be called the 'planned delay', because only one of the terminals needs to wait during its planned delay before broadcasting its packet. When the first packet of the round is broadcast (either by Alice or Bob with 0.5 probability), the other terminal has no reason to wait further and may broadcast its packet immediately (i.e. before its planned time delay has elapsed). Another important feature of this algorithm is that the second packet of each round can also be used for re-synchronization for the next round as depicted in Figure 9. Thus, it also allows for quickly detecting packet loss within a well-defined short time delay δ.

In our experiments, we have chosen a constant inter-round time which is $T = 0.2$ seconds. This setting was suitable for satisfying the required laptop movements. Thus, using our basic protocol with 1 message per secret bit, 16 seconds were needed to agree upon a 80-bit secret key. For smaller devices that can be moved more rapidly and with better antennas (i.e. more omnidirectional than our cards) that do not require turning the devices in every directions, smaller T values can be used.

Figure 10 displays the sequences T_A and T_B. We denote T_A the sequence of the 80 values that represent the time interval between two consecutive packets sent by A. Similarly, P_B is the sequence of the 80 values that represent the timing interval between two consecutive packets sent by B. This figure shows that both signals look random. We furthermore executed the significance test DoM and SoR on these sequences. The results, presented in Table 2, show that the probability α is much larger than 1% for both tests and therefore these two sequences are most likely indistinguishable. It appears then very difficult for an attacker to differentiate A's packets from $B's$ ones using timing analysis.

Table 2: Significance tests for Timing Analysis

Distance of means	0.82(0.22)
Sum of ranks	0.96(0.053)

5.3 Energy consumption considerations

In this section, we compare the power consumption of our scheme with the power consumption of a Diffie-Hellman based pairing. For the purpose of this comparison, we assume that the two devices being paired are sensors using TinyOS and that the size of the generated shared key is 72 bits.

With our scheme, each device must receive and send 36 packets. Considering that a TinyOS packet that has a header size of 7 bytes [11], each device must send and receive 2016 bits ($36 * 8 * 7$) (as explained in Section 4, in our basic protocol, packets do not have to contain any payload). However transmitting one bit consumes about as much power as executing 800-1000 intructions [8, 11]. Receiving one bit consumes about half as much power as sending one bit. As a result, our protocol consumes as much energy as the execution of about $2.72 * 10^6$ instructions ($2016 * 900 + 2016 * 450$).

In comparison, with a Diffie-Hellman based pairing protocol, each device needs to exchange their Diffie-Hellman public component (i.e. g^x, where x is the device's private key). A security equivalent to 72 bits requires to select a modulus of 1024 bits and an exponent of 130 bits [12]. As a result, the device's Diffie-Hellman public component is 1024-bit long. Since the maximum number of payload bits in a TinyOS packet is 232, each device must send (and receive) 5 packets. Therefore, the total number of bits sent and received is 1304 bits: 4 packets containing 232 bits and 1 packet containing 96 bits. This consumes as much energy as the execution of $1.76 * 10^6$ instructions ($1304 * 900 + 1304 * 450$). Upon reception of the other party's public component, each device has to exponentiate it with its Diffie-Hellman private key. Exponentiating using the Montgomery algorithm requires $3 * l * (l + 1) * (t + 1)$ single-precision multiplications, where l is the size of the modulus and t the size of the exponent [13]. With $l = 1024$ and $t = 130$, each device must perform $4.12 * 10^8$ single-precision multiplications. In conclusion, the total power consumed by each device is therefore equivalent to the power consumed by the execution of $1.76 * 10^6 + 4.12 * 10^8$ instructions. This cost is about *100 times* larger than the cost of our scheme.

The bandwidth cost of the Diffie-Hellman based solution could be significantly decreased with elliptic curves. In fact, the security of a 1024-bit Diffie-Hellman key exchange is equivalent to the security of a 135-bit Elliptic Curve Diffie-Hellman (EC-DH) key exchange [12]. Therefore only one packet would be necessary to be exchanged by the two devices. This would reduce the energy cost due to communication by 5. However, as shown in [3], EC-DH key derivation cost is even more expensive than regular DH key derivation. Therefore the total energy cost would still be much higher than the cost of our scheme.

Note the power consumption of imprinting and PIN-based solutions (see Section 2) is much lower than the power consumption of our scheme. However, these schemes are not

really comparable to ours since they both require additional interfaces which may increase the cost of devices.

6 Conclusion

In this paper we presented a novel secure pairing scheme for CPU-constrained devices without a need for special hardware or interfaces. Using an existing communication channel such as 802.11 or 802.15.4, two devices that are close to each other can agree on a secret key using an algorithm that does not depend on CPU-intensive operations. On the other hand, user assistance is required for *shaking* the devices during key agreement in order to preserve key secrecy.

One limitation of our scheme is that it is specific to random media access technologies. For example, it is not suitable for TDMA-based protocols and, therefore, cannot be used with Bluetooth devices. Our scheme requires CSMA-based systems, such as 802.11 or 802.15.4 (an emerging Wireless PAN (WPAN) technology, designed for low power sensors). Another noticeable limitation is that it requires that the transmission power of both devices be similar. This was the case with the 802.11 devices that we used for our experimentations. However, for some wireless technologies, a power control protocol might be required to adjust the transmission power accordingly.

Apart from these weaknesses, we believe that this is a very refreshing approach and offers a new perspective to wireless communication system security. To our knowledge, this is the first practical pairing protocol that does not rely on expensive cryptographic operations or require a secure out-of band channel.

Objects with microprocessors and wireless transmitters surround us. Today's users are more and more technology and security-aware. Almost all users today learned that a system access password should contain non-alphanumeric characters. We have learned (or are forced to learn) how to handle computer viruses. Technology and information security have become part of our everyday lives. Thus, we believe that future users can also learn when *two small devices must be shaken well before secure use*. Note that this is actually a very common protocol that we already execute in our everyday lives. For example, orange juice or shaving cream bottles are universally shaken/moved before usage. This is now a well-known and quite a natural "protocol". Furthermore it is commonly accepted that it is the responsability and interest of the consumers to perform this shaking operation properly. Similarly, in our case by shaking the devices well, the user can make sure that the two devices are paired securely.

References

[1] B. Alpern and F. Schneider. Key exchange using 'keyless cryptography. *Information processing letters*, 16(2):79–82, February 1983.

[2] J.-S. Coron, P. Kocher, and N. D. Statistics and Secret Leakage. In *Financial Cryptography*, 2000.

[3] W. Dai. Speed benchmarks for various ciphers and hash functions. URL:http://www.eskimo.com/~weidai/.

[4] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[5] C. Gehrmann and K. Nyberg. Enhancements to bluetooth baseband security. In *Nordsec'01*, Kopenhagen, Denmark, November 2001.

[6] I. Gibra. *Probability and Statistical Inference for Scientists and Engineers.* Prentice-Hall, 1973.

[7] S. Goldwasser and M. Bellare. Lectures notes in cryptography. URL:http://www.cs.ucsd.edu/users/mihir/papers/gb.html.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[9] J.-H. Hoepman. Ephemeral pairing in anonymous networks. Available at: http://www.cs.kun.nl/~jhh/publications/anon-pairing.pdf.

[10] J.-H. Hoepman. The ephemeral pairing problem. In *8th Int. Conf. Financial Cryptography*, pages 212–226, Key West, Florida, February 9-12 2004.

[11] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[12] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[13] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press series on discrete mathematics and its applications. 1997. ISBN 0-8493-8523-7.

[14] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Coomunications of the ACM*, 21:120–126, 1978.

[15] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, 1999.

[16] R. Want. New horizons for mobile computing. Keynote Speech at PerCom'03, 2003. URL:http://www.percom.org/percom_2003/slides/.

A The difference of means test[7]

The Difference of Means test (DoM) is a significance test that returns a probability α that an observed difference in the means of two sequences X and Y rises by chance, assuming that X and Y were generated by the same source.

By the virtue of the Central Limit Theorem, the experimental averages of X and Y (respectivily $\bar{X}$ and $\bar{Y}$) are approximately Gaussian, independent, of expectations $\{\mu[X], \mu[y]\}$ and variances $\{\sigma[X]^2/n[X], \sigma[Y]^2/n[Y]\}$; where $n[U]$ denotes the number of elements in the set U. We can therefore compute the reduced Gaussian variable:

$$\epsilon = \frac{\bar{X} - \bar{Y}}{\sqrt{\sigma[X]^2/n[X] + \sigma[Y]^2/n[Y]}}$$

and look-up its corresponding value in the Cumulative Density Function (CDF) Gaussian table (Table 3) which yields the hypothesis' significance α representing the probability that the reduced deviation will equate or exceed in absolute value a given ϵ.

Table 3: CDF Gaussian table

α	0.000	0.010	0.020	0.030	0.040	0.050	0.060	0.070	0.080	0.090
0.00	∞	2.576	2.326	2.170	2.054	1.960	1.881	1.812	1.751	1.695
0.10	1.645	1.598	1.555	1.514	1.476	1.440	1.405	1.327	1.341	1.311
0.20	1.282	1.254	1.227	1.200	1.175	1.150	1.126	1.103	1.080	1.058
0.30	1.036	1.015	0.994	0.974	0.954	0.935	0.915	0.896	0.878	0.860
0.40	0.842	0.824	0.806	0.789	0.772	0.755	0.739	0.722	0.706	0.690
0.50	0.674	0.659	0.643	0.628	0.613	0.598	0.583	0.568	0.553	0.539
0.60	0.524	0.510	0.496	0.482	0.468	0.454	0.440	0.426	0.412	0.399
0.70	0.385	0.372	0.358	0.345	0.332	0.319	0.305	0.292	0.279	0.266
0.80	0.253	0.240	0.228	0.215	0.202	0.189	0.176	0.164	0.151	0.138
0.90	0.126	0.113	0.100	0.088	0.075	0.063	0.050	0.038	0.025	0.013

α is obtained by adding the two numbers appearing in the margins (for instance: for $\epsilon = 1.960$, $table[\epsilon] = 0.000 + 0.05 = 0.05$).

[7] This appendix has been extracted from [2].

Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399